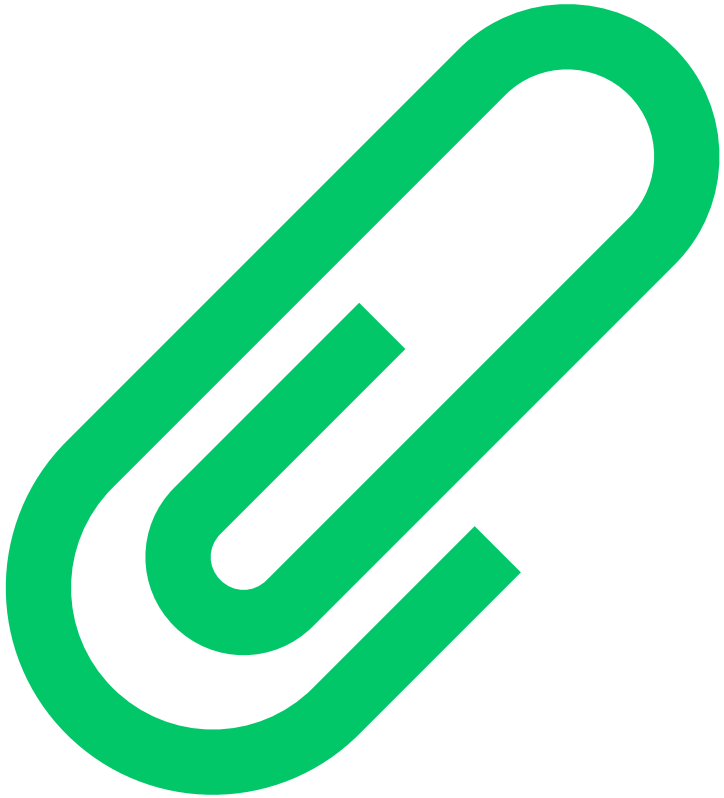
FaS 13차 세미나

파일 복호화 실습 -Session

20192233 박진철

CONTENTS

1. 첨부 파일 분석
2. 암호화 과정 분석
3. 첨부 파일 복호화
4. 복호화된 파일 확인



1. 첨부 파일 분석

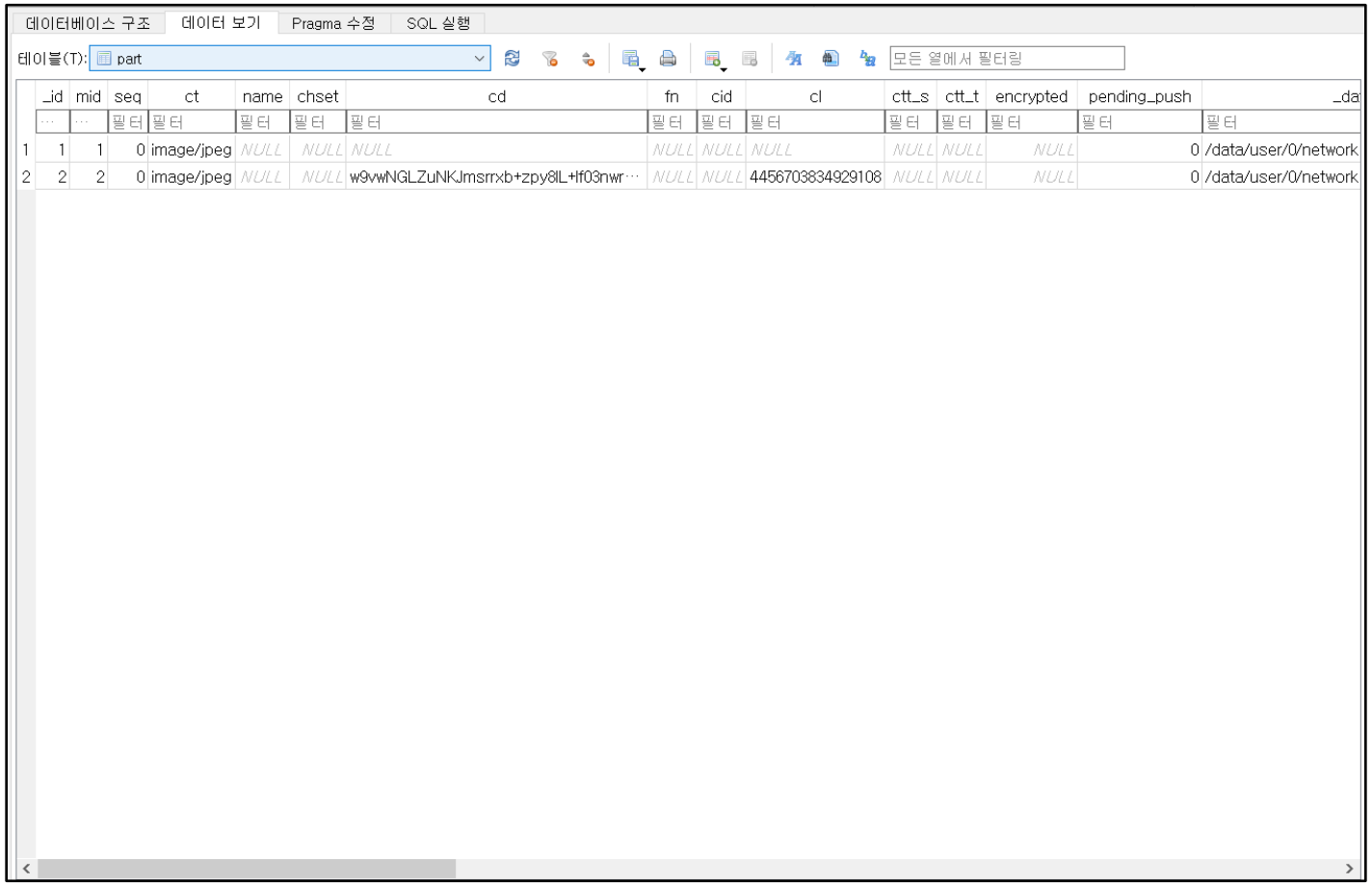
1. 첨부 파일 분석

테이블(T): sms_fts

	body	thread_id
	필터	필터
1	hello world!!	1
2	very good	1
3	testing...	1
4	godisnowhere	1
5	the quick brown fox jumps over the lazy dog	1
6	help	2
7	hello world!	2
8	hihi	1
9	kali linux Metasploit library is strong	1

👉 메시지 내용까지 복호화 성공

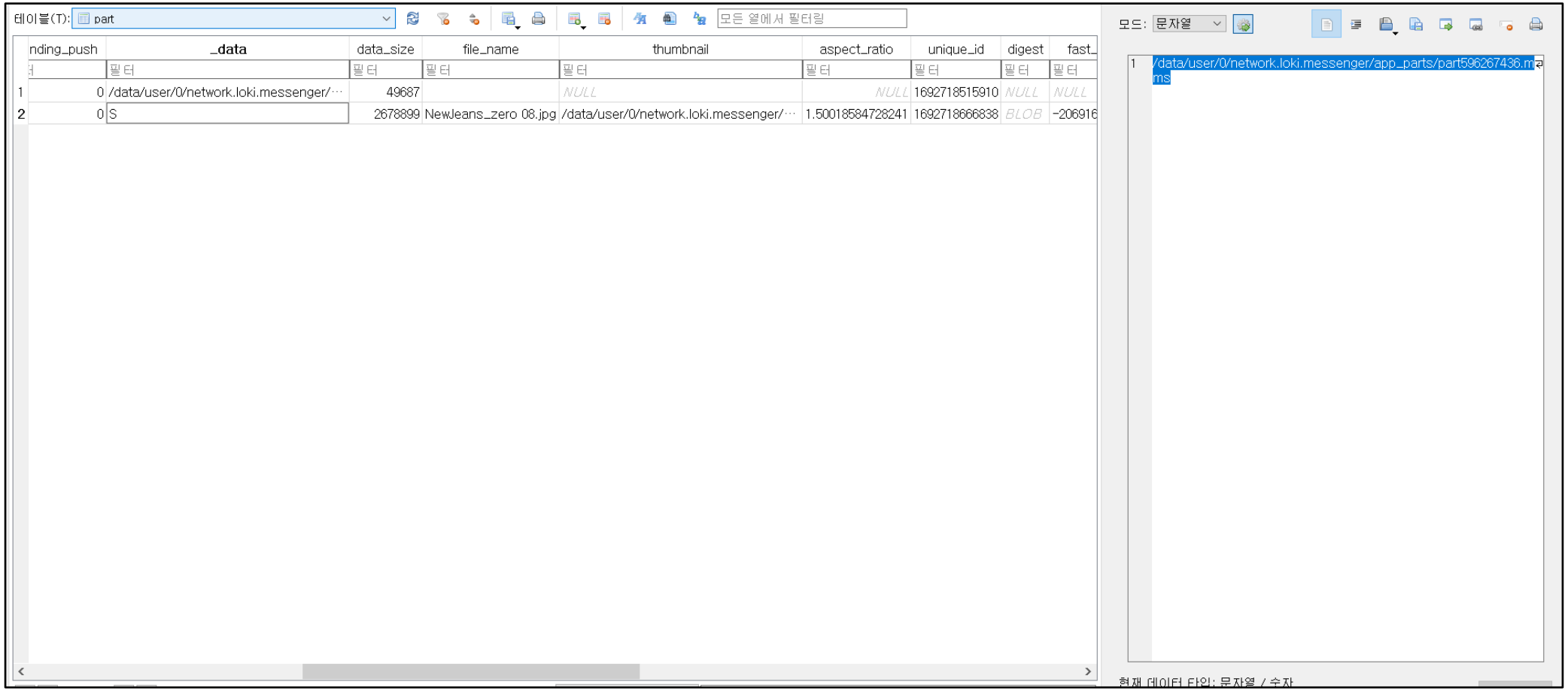
1. 첨부 파일 분석



	_id	mid	seq	ct	name	chset	cd	fn	cid	cl	ctt_s	ctt_t	encrypted	pending_push	_da
	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터
1	1	1	0	image/jpeg	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	/data/user/0/network
2	2	2	0	image/jpeg	NULL	NULL	w9vwNGLZuNKJmsrrxb+zpy8lL+lf03nwr···	NULL	NULL	4456703834929108	NULL	NULL	NULL	0	/data/user/0/network

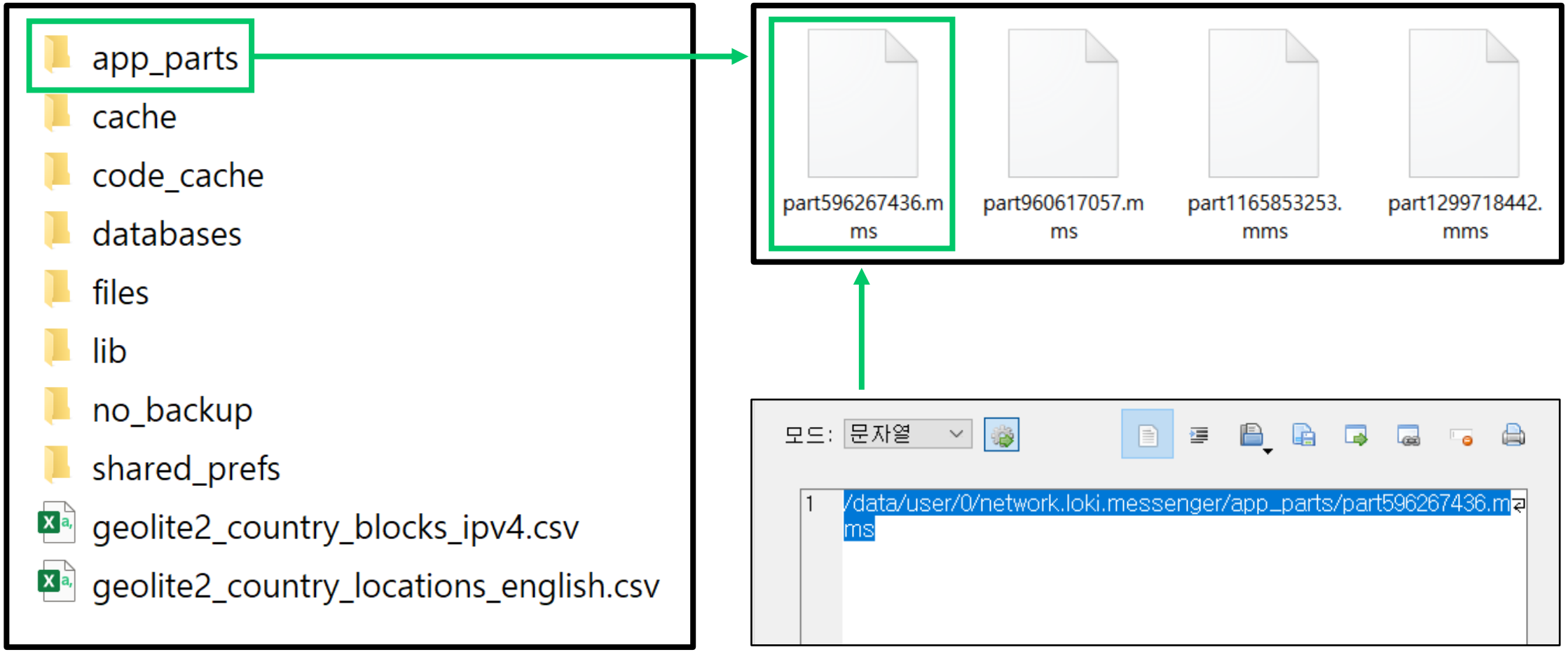
☞ 첨부된 파일들은 암호화되어 있음

1. 첨부 파일 분석



☞ 첨부된 파일들은 암호화되어 있음

1. 첨부 파일 분석



☞ 첨부된 파일들은 암호화되어 있음

2. 암호화 과정 분석

2. 암호화 과정 분석

```
public static final String DIRECTORY = "parts";
```

```
private DataInfo setAttachmentData(InputStream inputStream) throws MmsException {
    try {
        return setAttachmentData(File.createTempFile(TABLE_NAME, ".mms", this.context.getDir(DIRECTORY, 0)), inputStream);
    } catch (IOException e) {
        throw new MmsException(e);
    }
}
```

```
public static final String TABLE_NAME = "part";
```

```
private DataInfo setAttachmentData(File file, InputStream inputStream) throws MmsException {
    try {
        Pair<byte[], OutputStream> createFor = ModernEncryptingPartOutputStream.createFor(this.attachmentSecret, file, false);
        return new DataInfo(file, Util.copy(inputStream, (OutputStream) createFor.second), (byte[]) createFor.first);
    } catch (IOException e) {
        throw new MmsException(e);
    }
}
```

```
private DataInfo(File file, long j, byte[] bArr) {
    this.file = file;
    this.length = j;
    this.random = bArr;
}
```

☞ .mms 파일을 만드는 클래스로 이동

2. 암호화 과정 분석

```
private DataInfo setAttachmentData(File file, InputStream inputStream) throws MmsException {
    try {
        Pair<byte[], OutputStream> createFor = ModernEncryptingPartOutputStream.createFor(this.attachmentSecret, file, false);
        return new DataInfo(file, Util.copy(inputStream, (OutputStream) createFor.second), (byte[]) createFor.first);
    } catch (IOException e) {
```

```
public static Pair<byte[], OutputStream> createFor(AttachmentSecret attachmentSecret, File file, boolean z) throws IOException {
    byte[] bArr = new byte[32];
    new SecureRandom().nextBytes(bArr);
    try {
        Mac mac = Mac.getInstance("HmacSHA256");
        mac.init(new SecretKeySpec(attachmentSecret.getModernKey(), "HmacSHA256"));
        FileOutputStream fileOutputStream = new FileOutputStream(file);
        byte[] doFinal = mac.doFinal(bArr);
        Cipher cipher = Cipher.getInstance("AES/CTR/NoPadding");
        cipher.init(1, new SecretKeySpec(doFinal, "AES"), new IvParameterSpec(new byte[16]));
        if (z) {
            fileOutputStream.write(bArr);
        }
        return new Pair<>(bArr, new CipherOutputStream(fileOutputStream, cipher));
    } catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e) {
        throw new AssertionError(e);
    }
}
```

👉 createfor()라는 메소드를 통해 파일을 암호화

2. 암호화 과정 분석

```
public static Pair<byte[], OutputStream> createFor(AttachmentSecret attachmentSecret, File file, boolean z) throws IOException {
    byte[] bArr = new byte[32];
    new SecureRandom().nextBytes(bArr);
    try {
        Mac mac = Mac.getInstance("HmacSHA256");
        mac.init(new SecretKeySpec(attachmentSecret.getModernKey(), "HmacSHA256"));
        FileOutputStream fileOutputStream = new FileOutputStream(file);
        byte[] doFinal = mac.doFinal(bArr);
        Cipher cipher = Cipher.getInstance("AES/CTR/NoPadding");
        cipher.init(1, new SecretKeySpec(doFinal, "AES"), new IvParameterSpec(new byte[16]));
        if (z) {
            fileOutputStream.write(bArr);
        }
        return new Pair<>(bArr, new CipherOutputStream(fileOutputStream, cipher));
    } catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e) {
        throw new AssertionError(e);
    }
}
```

👉 createfor()라는 메소드를 통해 파일을 암호화

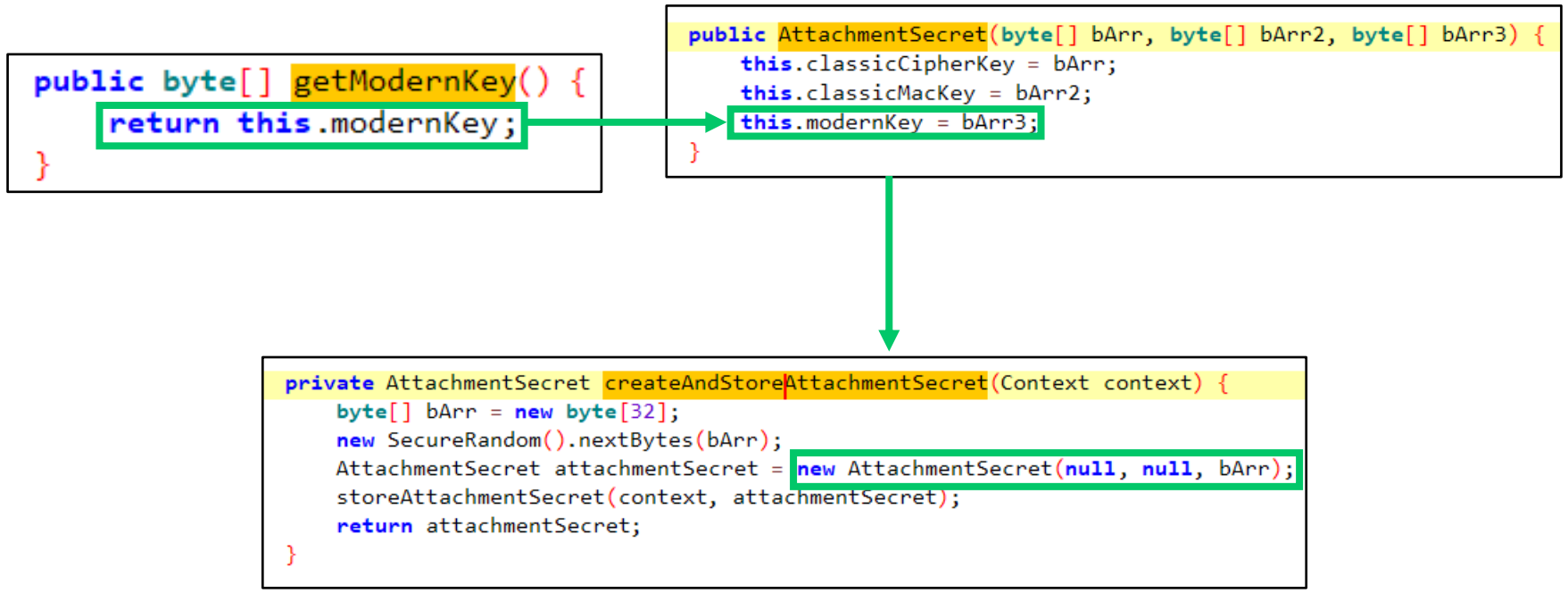
2. 암호화 과정 분석

```
public static Pair<byte[], OutputStream> createFor(AttachmentSecret attachmentSecret, File file, boolean z) throws IOException {
    byte[] bArr = new byte[32];
    new SecureRandom().nextBytes(bArr);
    try {
        Mac mac = Mac.getInstance("HmacSHA256");
        mac.init(new SecretKeySpec(attachmentSecret.getModernKey(), "HmacSHA256"));
        FileOutputStream fileOutputStream = new FileOutputStream(file);
        byte[] doFinal = mac.doFinal(bArr);
        Cipher cipher = Cipher.ge
        cipher.init(1, new Secret                                    terSpec(new byte[16]));
        if (z) {
            fileOutputStream.writ
        }
        return new Pair<>(bArr, new CipherOutputStream(fileOutputStream, cipher));
    } catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e) {
        throw new AssertionError(e);
    }
}
```

```
public byte[] getModernKey() {
    return this.modernKey;
}
```

👉 getModernKey()메소드를 통해 modernKey를 리턴받음

2. 암호화 과정 분석



☞ modernKey는 AttachmentSecret()메소드를 통해 변경

2. 암호화 과정 분석

```
private AttachmentSecret createAndStoreAttachmentSecret(Context context) {
    byte[] bArr = new byte[32];
    new SecureRandom().nextBytes(bArr);
    AttachmentSecret attachmentSecret = new AttachmentSecret(null, null, bArr);
    storeAttachmentSecret(context, attachmentSecret);
    return attachmentSecret;
}
```

```
public synchronized AttachmentSecret getOrCreateAttachmentSecret() {
    AttachmentSecret attachmentSecret = this.attachmentSecret;
    if (attachmentSecret != null) {
        return attachmentSecret;
    }
    String attachmentUnencryptedSecret = TextSecurePreferences.CC.getAttachmentUnencryptedSecret(this.context);
    String attachmentEncryptedSecret = TextSecurePreferences.CC.getAttachmentEncryptedSecret(this.context);
    if (attachmentUnencryptedSecret != null) {
        this.attachmentSecret = getUnencryptedAttachmentSecret(this.context, attachmentUnencryptedSecret);
    } else if (attachmentEncryptedSecret != null) {
        this.attachmentSecret = getEncryptedAttachmentSecret(attachmentEncryptedSecret);
    } else {
        this.attachmentSecret = createAndStoreAttachmentSecret(this.context);
    }
    return this.attachmentSecret;
}
```

👉 createAndStore…Secret()메소드를 사용하는 곳으로 이동

2. 암호화 과정 분석

```
public synchronized AttachmentSecret getOrCreateAttachmentSecret() {
    AttachmentSecret attachmentSecret = this.attachmentSecret;
    if (attachmentSecret != null) {
        return attachmentSecret;
    }
    String attachmentUnencryptedSecret = TextSecurePreferences.CC.getAttachmentUnencryptedSecret(this.context);
    String attachmentEncryptedSecret = TextSecurePreferences.CC.getAttachmentEncryptedSecret(this.context);
    if (attachmentUnencryptedSecret != null) {
        this.attachmentSecret = getUnencryptedAttachmentSecret(this.context, attachmentUnencryptedSecret);
    } else if (attachmentEncryptedSecret != null) {
        this.attachmentSecret = getEncryptedAttachmentSecret(attachmentEncryptedSecret);
    } else {
        this.attachmentSecret = createAndStoreAttachmentSecret(this.context);
    }
    return this.attachmentSecret;
}
```

```
public static String getAttachmentUnencryptedSecret(Context context) {
    return TextSecurePreferences.Companion.getAttachmentUnencryptedSecret(context);
}
```

```
public final String getAttachmentUnencryptedSecret(Context context) {
    Intrinsics.checkNotNullParameter(context, "context");
    return getStringPreference(context, "pref_attachment_unencrypted_secret", null);
}
```

```
public static String getAttachmentEncryptedSecret(Context context) {
    return TextSecurePreferences.Companion.getAttachmentEncryptedSecret(context);
}
```

```
public final String getAttachmentEncryptedSecret(Context context) {
    Intrinsics.checkNotNullParameter(context, "context");
    return getStringPreference(context, "pref_attachment_encrypted_secret", null);
}
```

```
<string name="pref_attachment_encrypted_secret">
{"data":"Z4mceWOm1AWwe6nWPJ15wRsIPC4PR8ys010gzX1EG79oCC0SVX2TwQsgjjOMtcikhrTEwQcRVihJiDIebhMPUiW5m74RIZ3Cjz32xrhcfcV4rYTec4NMiQAcSFvk/0v9yGhBFNzGQI7PQqcFAb/53fxETax4vc0g","iv":"FDp0uyvr+KbzML+1"}</string>
```

👉 xml파일 내에 있는 pref_attach…를 추출

2. 암호화 과정 분석

```
public synchronized AttachmentSecret getOrCreateAttachmentSecret() {
    AttachmentSecret attachmentSecret = this.attachmentSecret;
    if (attachmentSecret != null) {
        return attachmentSecret;
    }
    String attachmentUnencryptedSecret = TextSecurePreferences.CC.getAttachmentUnencryptedSecret(this.context);
    String attachmentEncryptedSecret = TextSecurePreferences.CC.getAttachmentEncryptedSecret(this.context);
    if (attachmentUnencryptedSecret != null) {
        this.attachmentSecret = getUnencryptedAttachmentSecret(this.context, attachmentUnencryptedSecret);
    } else if (attachmentEncryptedSecret != null) {
        this.attachmentSecret = getEncryptedAttachmentSecret(attachmentEncryptedSecret);
    } else {
        this.attachmentSecret = createAndStoreAttachmentSecret(this.context);
    }
    return this.attachmentSecret;
}
```

```
private AttachmentSecret getEncryptedAttachmentSecret(String str) {
    if (Build.VERSION.SDK_INT < 23) {
        throw new AssertionError("OS downgrade not supported. KeyStore sealed data exists on platform < M!");
    }
    return AttachmentSecret.fromString(new String(KeyStoreHelper.unseal(KeyStoreHelper.SealedData.fromString(str))));
}
```

👉 getEncrypted…Secret()를 통해 키를 생성

2. 암호화 과정 분석

```
private AttachmentSecret getEncryptedAttachmentSecret(String str) {
    if (Build.VERSION.SDK_INT < 23) {
        throw new AssertionError("OS downgrade not supported. KeyStore sealed data exists on platform < M!");
    }
    return AttachmentSecret.fromString(new String(KeyStoreHelper.unseal(KeyStoreHelper.SealedData.fromString(str))));
}
```

```
public static byte[] unseal(SealedData sealedData) {
    byte[] doFinal;
    SecretKey keyStoreEntry = getKeyStoreEntry();
    try {
        synchronized (CipherUtil.CIPHER_LOCK) {
            Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
            cipher.init(2, keyStoreEntry, new GCMParameterSpec(128, sealedData.f990iv));
            doFinal = cipher.doFinal(sealedData.data);
        }
        return doFinal;
    } catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException | BadPaddingException | IllegalBlockSizeException | NoSuchPaddingException e) {
        throw new AssertionError(e);
    }
}
```

☞ unseal()메소드는 데이터 베이스 키 생성과 동일

2. 암호화 과정 분석

```java
public static Pair<byte[], OutputStream> createFor(AttachmentSecret attachmentSecret, File file, boolean z) throws IOException {
    byte[] bArr = new byte[32];
    new SecureRandom().nextBytes(bArr);
    try {
        Mac mac = Mac.getInstance("HmacSHA256");
        mac.init(new SecretKeySpec(attachmentSecret.getModernKey(), "HmacSHA256"));
        FileOutputStream fileOutputStream = new FileOutputStream(file);
        byte[] doFinal = mac.doFinal(bArr);
        Cipher cipher = Cipher.getInstance("AES/CTR/NoPadding");
        cipher.init(1, new SecretKeySpec(doFinal, "AES"), new IvParameterSpec(new byte[16]));
        if (z) {
            fileOutputStream.write(bArr);
        }
        return new Pair<>(bArr, new CipherOutputStream(fileOutputStream, cipher));
    } catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e) {
        throw new AssertionError(e);
    }
}
```

👉 Key: 데이터베이스 키와 동일한 알고리즘으로 추출한 modernKey

👉 Data: 32바이트의 랜덤한 값

👉 Hmac까지의 과정을 통해 AES의 키를 얻음

2. 암호화 과정 분석

```java
public static Pair<byte[], OutputStream> createFor(AttachmentSecret attachmentSecret, File file, boolean z) throws IOException {
    byte[] bArr = new byte[32];
    new SecureRandom().nextBytes(bArr);
    try {
        Mac mac = Mac.getInstance("HmacSHA256");
        mac.init(new SecretKeySpec(attachmentSecret.getModernKey(), "HmacSHA256"));
        FileOutputStream fileOutputStream = new FileOutputStream(file);
        byte[] doFinal = mac.doFinal(bArr);
        Cipher cipher = Cipher.getInstance("AES/CTR/NoPadding");
        cipher.init(1, new SecretKeySpec(doFinal, "AES"), new IvParameterSpec(new byte[16]));
        if (z) {
            fileOutputStream.write(bArr);
        }
        return new Pair<>(bArr, new CipherOutputStream(fileOutputStream, cipher));
    } catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e) {
        throw new AssertionError(e);
    }
}
```

- 👉 Key: modernKey를 HmacSHA256에 넣은 값
- 👉 Data: 입력받은 파일(첨부파일)
- 👉 IV: 0

👉 AES-CTR을 이용해 파일을 암호화

2. 암호화 과정 분석

```
InPut: pref_attachment_encrypted_secret_data,
       pref_attachment_encrypted_secret_IV
OutPut: Dec_File

Data ← pref_attachment_encrypted_secret_data
IV ← pref_attachment_encrypted_secret_IV
modernKey ← get_database_key(data, android_Key, IV)
AES_Key ← HmacSHA256(modernKey, random[32])
Data ← mms file
IV ← 0
Dec_File ← AES_CTR_Dec(AES_Key, Data, IV)
```

👉 파일 복호화 Psudo-code

3. 첨부파일 복호화

3. 첨부파일 복호화

```
InPut: pref_attachment_encrypted_secret_data,
       pref_attachment_encrypted_secret_IV
OutPut: Dec_File

Data ← pref_attachment_encrypted_secret_data
IV ← pref_attachment_encrypted_secret_IV
modernKey ← get_database_key(data, android_Key, IV)
AES_Key ← HmacSHA256(modernKey, random[32])
Data ← mms file
IV ← 0
Dec_File ← AES_CTR_Dec(AES_Key, Data, IV)
```

👉 modernKey값은 어떻게 얻는가?
👉 random값은 어떻게 얻는가?

3. 첨부파일 복호화

```
private AttachmentSecret getEncryptedAttachmentSecret(String str) {
    if (Build.VERSION.SDK_INT < 23) {
        throw new AssertionError("OS downgrade not supported. KeyStore sealed data exists on platform < M!");
    }
    return AttachmentSecret.fromString(new String(KeyStoreHelper.unseal(KeyStoreHelper.SealedData.fromString(str))));
}
```

```
public static AttachmentSecret fromString(String str) {
    try {
        return (AttachmentSecret) JsonUtil.fromJson(str, AttachmentSecret.class);
    } catch (IOException e) {
        throw new AssertionError(e);
    }
}
```

```
public static <T> T fromJson(String str, Class<T> cls) throws IOException {
    return (T) objectMapper.readValue(str, cls);
}
```

👉 modernKey는 복호화 후, 그 문단에서 modernkey를 획득

3. 첨부파일 복호화

<string name="pref_attachment_encrypted_secret">
{"data":"Z4mceWOm1AWwe6nWPJ15wRsIPC4PR8ys010gzX1EG79oCCOSVX2TwQsgjjOMtcikhrTEwQcRVihJiDIebhMPUiW5m74RIZ3Cjz32xrhcfcV4rYTec4NMiQAcSFvk/0v9yGhBFNzGQI7PQqcFAb/53fxETax4vc0g","iv":"FDp0uyvr+KbzML+1"}</string>

InPut: pref_attachment_encrypted_secret_data,
pref_attachment_encrypted_secret_IV
OutPut: Dec_File

Data ← pref_attachment_encrypted_secret_data
IV ← pref_attachment_encrypted_secret_IV
modernKey ← get_database_key(data, android_Key, IV)
AES_Key ← HmacSHA256(modernKey, random[32])
Data ← mms file
IV ← 0
Dec_File ← AES_CTR_Dec(AES_Key, Data, IV)

```
Session 데이터베이스 키 추출 코드
"""
from Crypto.Cipher import AES
import base64

#data, iv, key 값을 입력
encrypted_secret_data=base64.b64decode("Z4mceWOm1AWwe6nWPJ15wRslPC4PR8ysO1OgzX1EG7
encrypted_secret_iv=base64.b64decode("FDp0uyvr+KbzML+1".encode())
Key=b'\x3F\x9F\x47\xA9\xDB\x8F\x51\x23\xB1\xD1\x85\xA7\x5B\x01\x93\xE1'

#얻은 데이터에서 필요한 값만 추출
data=encrypted_secret_data
iv=encrypted_secret_iv

#AES GCM 모드로 복호화
cipher=AES.new(Key, AES.MODE_GCM, iv)
dec_key=cipher.decrypt(data)

#키 출력
sql_key=dec_key.hex()
print(sql_key)
```

👉 modernKey는 복호화 후, 그 문단에서 modernkey를 획득

3. 첨부파일 복호화

InPut: pref_attachment_encrypted_secret_data,
pref_attachment_encrypted_secret_IV
OutPut: Dec_File

Data ← pref_attachment_encrypted_secret_data
IV ← pref_attachment_encrypted_secret_IV
modernKey ← get_database_key(data, android_Key, IV)
AES_Key ← HmacSHA256(modernKey, random[32])
Data ← mms file
IV ← 0
Dec_File ← AES_CTR_Dec(AES_Key, Data, IV)

```
7b22636c617373696343697068657 24b6579223a6e756c6c2c22636c61737369634d61634b6579223a6e756c6c2c226d6f6465726e4b6579223a224b7446437442537337683 54e41575563487
544547a686d35654951395577562f6b2f374f50376761735945227de5e503171ce3cf15a2a6ffcf8db6fb71
```

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	7B	22	63	6C	61	73	73	69	63	43	69	70	68	65	72	4B	{"classicCipherK
00000010	65	79	22	3A	6E	75	6C	6C	2C	22	63	6C	61	73	73	69	ey":null,"classi
00000020	63	4D	61	63	4B	65	79	22	3A	6E	75	6C	6C	2C	22	6D	cMacKey":null,"m
00000030	6F	64	65	72	6E	4B	65	79	22	3A	22	4B	74	46	43	74	odernKey":"KtFCt
00000040	42	53	73	37	68	35	4E	41	57	55	63	48	75	44	54	7A	BSs7h5NAWUcHuDTz
00000050	68	6D	35	65	49	51	39	55	77	56	2F	6B	2F	37	4F	50	hm5eIQ9UwV/k/7OP
00000060	37	67	61	73	59	45	22	7D	E5	E5	03	17	1C	E3	CF	15	7gasYE"}åå...ãÏ.
00000070	A2	A6	FF	CF	8D	B6	FB	71									¢¦ÿÏ.¶ûq

👉 modernKey는 복호화 후, 그 문단에서 modernkey를 획득

3. 첨부파일 복호화

```
public static Pair<byte[], OutputStream> createFor(AttachmentSecret attachmentSecret, File file, boolean z) throws IOException {
    byte[] bArr = new byte[32];
    new SecureRandom().nextBytes(bArr);
    try {
        Mac mac = Mac.getInstance("HmacSHA256");
        mac.init(new SecretKeySpec(attachmentSecret.getModernKey(), "HmacSHA256"));
        FileOutputStream fileOutputStream = new FileOutputStream(file);
        byte[] doFinal = mac.doFinal(bArr);
        Cipher cipher = Cipher.getInstance("AES/CTR/NoPadding");
        cipher.init(1, new SecretKeySpec(doFinal, "AES"), new IvParameterSpec(new byte[16]));
        if (z) {
            fileOutputStream.write(bArr);
        }
        return new Pair<>(bArr, new CipherOutputStream(fileOutputStream, cipher));
    } catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e) {
```

```
private DataInfo setAttachmentData(File file, InputStream inputStream) throws MmsException {
    try {
        Pair<byte[], OutputStream> createFor = ModernEncryptingPartOutputStream.createFor(this.attachmentSecret, file, false);
        return new DataInfo(file, Util.copy(inputStream, (OutputStream) createFor.second), (byte[]) createFor.first);
    } catch (IOException e) {
        throw new MmsException(e);
    }
}
```

👉 random값은 _data_random에 들어있음

3. 첨부파일 복호화

```
private DataInfo setAttachmentData(File file, InputStream inputStream) throws MmsException {
    try {
        Pair<byte[], OutputStream> createFor = ModernEncryptingPartOutputStream.createFor(this.attachmentSecret, file, false);
        return new DataInfo(file, Util.copy(inputStream, (OutputStream) createFor.second), (byte[]) createFor.first);
    } catch (IOException e) {
        throw new MmsException(e);
    }
}
```

```
public static class DataInfo {
    private final File file;
    private final long length;
    private final byte[] random;

    private DataInfo(File file, long j, byte[] bArr) {
        this.file = file;
        this.length = j;
        this.random = bArr;
    }
}
```

👉 random값은 _data_random에 들어있음

3. 첨부파일 복호화

```
public static final String DATA_RANDOM = "data_random";
```

```
private AttachmentId insertAttachment(long j, Attachment attachment, boolean z) throws MmsException {
    DataInfo dataInfo;
    Pair<Integer, Integer> dimensions;
    String str = TAG;
    Log.m213d(str, "Inserting attachment for mms id: " + j);
    SQLiteDatabase writableDatabase = this.databaseHelper.getWritableDatabase();
    long currentTimeMillis = System.currentTimeMillis();
    if (attachment.getDataUri() != null) {
        dataInfo = setAttachmentData(attachment.getDataUri());
        Log.m213d(str, "Wrote part to file: " + dataInfo.file.getAbsolutePath());
    } else {
        dataInfo = null;
    }
    ContentValues contentValues = new ContentValues();
    contentValues.put(MMS_ID, Long.valueOf(j));
    contentValues.put(CONTENT_TYPE, attachment.getContentType());
    contentValues.put(TRANSFER_STATE, Integer.valueOf(attachment.getTransferState()));
    contentValues.put(UNIQUE_ID, Long.valueOf(currentTimeMillis));
    contentValues.put(CONTENT_LOCATION, attachment.getLocation());
    contentValues.put(DIGEST, attachment.getDigest());
    contentValues.put(CONTENT_DISPOSITION, attachment.getKey());
    contentValues.put("name", attachment.getRelay());
    contentValues.put(FILE_NAME, ExternalStorageUtil.getCleanFileName(attachment.getFileName()));
    contentValues.put(SIZE, Long.valueOf(attachment.getSize()));
    contentValues.put(FAST_PREFLIGHT_ID, attachment.getFastPreflightId());
    contentValues.put(VOICE_NOTE, Integer.valueOf(attachment.isVoiceNote() ? 1 : 0));
    contentValues.put(WIDTH, Integer.valueOf(attachment.getWidth()));
    contentValues.put(HEIGHT, Integer.valueOf(attachment.getHeight()));
    contentValues.put("quote", Boolean.valueOf(z));
    contentValues.put("caption", attachment.getCaption());
    contentValues.put("url", attachment.getUrl());
    if (dataInfo != null) {
        contentValues.put("_data", dataInfo.file.getAbsolutePath());
        contentValues.put(SIZE, Long.valueOf(dataInfo.length));
        contentValues.put(DATA_RANDOM, dataInfo.random);
    }
```

```
public static class DataInfo {
    private final File file;
    private final long length;
    private final byte[] random;

    private DataInfo(File file, long j, byte[] bArr) {
        this.file = file;
        this.length = j;
        this.random = bArr;
    }
}
```

👉 random값은 _data_random에 들어있음

3. 첨부파일 복호화

테이블(T): part

모든 열에서 필터링

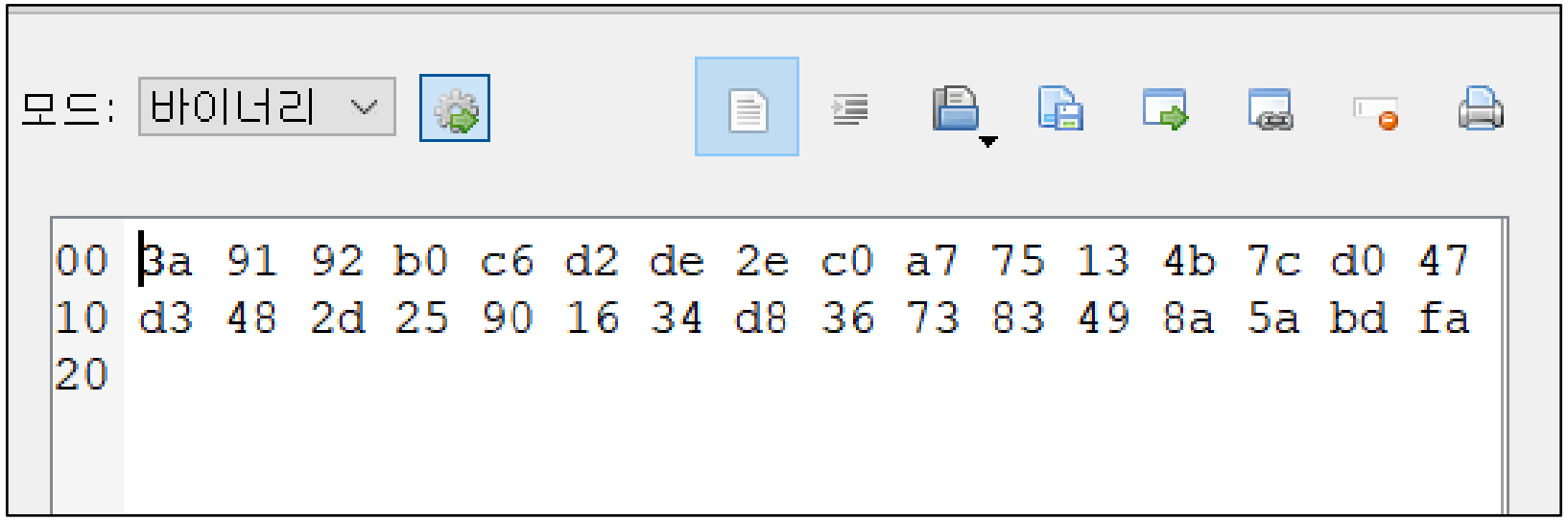
	thumbnail	aspect_ratio	unique_id	digest	fast_preflight_id	voice_note	data_random	thumbnail_random	quote	width	height	c
	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터	필
1	NULL	NULL	1692718515910	NULL	NULL	0	BLOB	NULL	0	1024	2160	N
2	'data/user/0/network.loki.messenger/···	1.50018584728241	1692718666838	BLOB	-206916536537918367	0	BLOB	BLOB	0	4096	2730	N

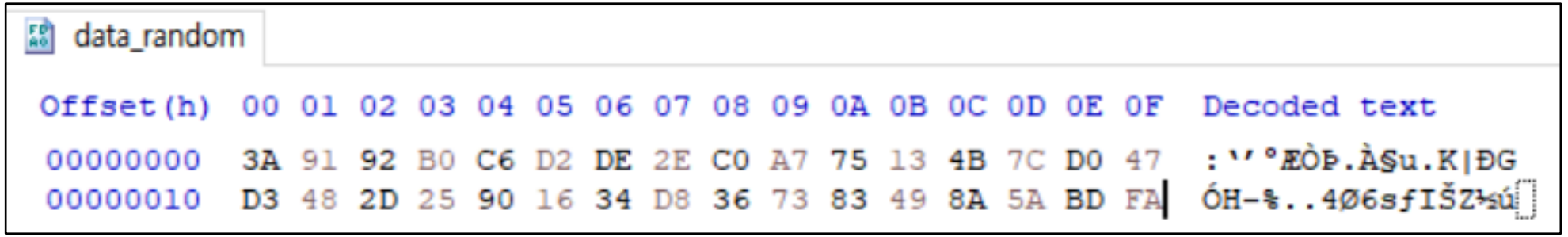
모드: 바이너리

```
00 3a 91 92 b0 c6 d2 de 2e c0 a7 75 13 4b 7c d0 47
10 d3 48 2d 25 90 16 34 d8 36 73 83 49 8a 5a bd fa
20
```

👉 random값은 _data_random에 들어있음

3. 첨부파일 복호화





☞ 해당 값을 data_random파일로 저장

3. 첨부파일 복호화

```python
from Crypto.Cipher import AES
from Crypto.Util import Counter
from binascii import hexlify
import base64
import hmac
import hashlib
import os

#정보 받기
dec_name=input("decrypt file name : ") #복호화된 파일의 이름 정하기
modernKey_v1=input("modernKey : ") #modernKey 입력
data_random_name=input("data_random file name : ") #data_random 값을 저장한 파일 경로 입력
mms_file_name=input("mms file name : ") #mms파일 경로 입력

#data_random 값 저장
with open(data_random_name, 'rb') as f:
    data_randon=f.read()
f.close()

#mms파일 값 저장
with open(mms_file_name, 'rb') as f:
    mms_file=f.read()
f.close()

#modernKey값을 base64로 저장
modernKey_v2 = modernKey_v1 + '=' * (4-len(modernKey_v1)%4)
modernKey=base64.b64decode(modernKey_v2)
```

InPut: pref_attachment_encrypted_secret_data,
pref_attachment_encrypted_secret_IV
OutPut: Dec_File

Data ← pref_attachment_encrypted_secret_data
IV ← pref_attachment_encrypted_secret_IV
modernKey ← get_database_key(data, android_Key, IV)
AES_Key ← HmacSHA256(modernKey, random[32])
Data ← mms file
IV ← 0
Dec_File ← AES_CTR_Dec(AES_Key, Data, IV)

3. 첨부파일 복호화

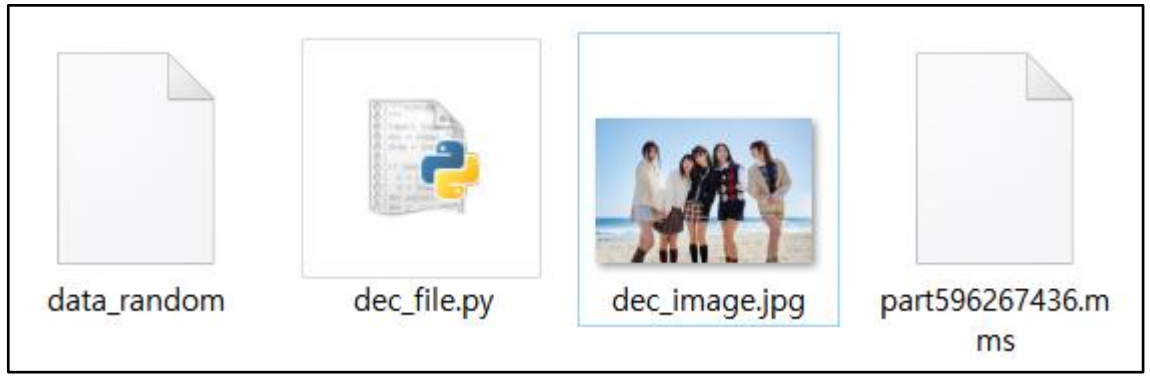
```
29  #HmacSHA256으로 키 생성
30  AES_Key=hmac.new(modernKey,data_randon,hashlib.sha256).digest()
31
32  #AES_CTR로 파일 복호화
33  specific_IV = 0
34  counter_value = Counter.new(128, initial_value=specific_IV)
35  cipher=AES.new(AES_Key, AES.MODE_CTR, counter=counter_value)
36  dec_file=cipher.decrypt(mms_file)
37
38  #복호화된 파일 저장
39  with open(os.path.join(os.path.dirname(mms_file_name), dec_name), 'wb') as f:
40      f.write(dec_file)
41
42  print("\ndone")
```

InPut: pref_attachment_encrypted_secret_data,
pref_attachment_encrypted_secret_IV
OutPut: Dec_File

Data ← pref_attachment_encrypted_secret_data
IV ← pref_attachment_encrypted_secret_IV
modernKey ← get_database_key(data, android_Key, IV)
AES_Key ← HmacSHA256(modernKey, random[32])
Data ← mms file
IV ← 0
Dec_File ← AES_CTR_Dec(AES_Key, Data, IV)

4. 복호화된 파일 확인

4. 복호화된 파일 확인




```
decrypt file name : dec_image
modernKey : KtFCtBSs7h5NAWUcHuDTzhm5eIQ9UwV/k/7OP7gasYE
data_random file name : c:\Users\SAMSUNG\Documents\decrypt\data_random
mms file name : c:\Users\SAMSUNG\Documents\decrypt\part596267436.mms

done
```

👉 복호화 성공!

THANKS TO WATCHING

20192233 박진철